Analyzing a Socio-Technical Visualization Tool Using Usability Inspection Methods

Erik Trainer[1] Stephen Quirk[1] Cleidson de Souza[2] David Redmiles[1]

[1]*Donald Bren School of Information and Computer Sciences*
University of California, Irvine
Irvine, CA, USA – 92667
[etrainer, squirk, redmiles]@ics.uci.edu

[2]*Faculdade de Computação*
Universidade Federal do Pará
Belém, PA, Brazil – 66075
cdesouza@ufpa.br

Abstract

Ariadne is a novel visualization tool that allows end users to explore the socio-technical relationships in software development projects. Essentially the visualization is a variant of a social network graph. It is based on the observation that dependencies between software components create dependencies between the developers implementing those components. This relationship emerged in our own and other researchers' field studies of software projects. Large software development projects require management of dependencies by managers and developers to ensure the smooth coordination of work. We sought to evaluate our visualization to assess its utility. Although we had some informal trials with potential end users, we sought a deeper analysis before further refinement of the tool and evaluation on a larger scale. Usability inspection methods provided one potential avenue. Moreover, such inspection methods yield a kind of rationale not directly derived from human subjects evaluations. We report on the application of these inspection methods and discuss the implications of their results in the context of usability evaluations for visual interfaces.

1. Introduction

It has been long recognized that breakdowns in communication and coordination efforts constitute a major problem in collaborative software development [6]. One of the reasons for these problems is the large number of dependencies among activities in the software development process and the dependencies among different software artifacts.

Parnas was one of the first researchers to recognize the relationship between software dependencies and coordination: he suggested that by reducing dependencies among software development artifacts, it is possible to reduce developers' dependencies on one another, creating a managerial advantage [13]. Nowadays, this is a well-known argument among researchers and practitioners.

Conversely, but also supporting this relationship between dependencies and coordination, Conway [5] postulated that the structure of a software system would reflect the communication needs of the people performing the work. That is, technical dependencies between components create a need for communication and coordination between developers, and similarly, dependencies between the development tasks are reflected in the product dependencies.

Ariadne's visualization was created with the aim of reducing the acknowledged gap between software dependencies and coordination by exploring socio-technical relationships to support software developers' activities. It combines source-code dependencies from static-call graphs and authorship from Configuration Management repositories to create a sociogram [17] that describes dependencies between developers through the code they write. This node-and-edge graph is calculated using a matrix multiplication method [3, 7].

During early development of the tool, we performed two key field studies, each 2-3 months in duration, that provided us insight into several types of communication and coordination problems in distributed software development projects. Of these issues, we derived several representative scenarios that revealed the types of dependency relationships managers and developers need to understand in order to coordinate their work [7]. Next, we designed an initial prototype and revised it after some early use and feedback from others.

In order to keep the visualization linked to human needs, we applied several usability inspection methods and cognitive theories to evaluate it against typical usage tasks we observed earlier. This paper reports on our inspections and discusses the results in the context of evaluation of visual interfaces.

The rest of this paper is structured as follows. Next, we present Ariadne's visualization. We follow up in sections 3 and 4 with the results of our evaluation and discuss the utility of inspection methods in early design. We conclude in section 5.

2. Visualization

Ariadne lays out dependency information in a table-based fashion, placing the most numerous data items along the longest screen dimension. Called code units occupy the x-axis and authors occupy the y-axis, with both ordered alphabetically by default. The visualization lays out code units organized by package, much how a programmer or manager might expect to see them in a development editor. To see dependencies within these packages (Figure 1), users can Ctrl+click on a package. Similarly they can click on classes to see method dependencies.

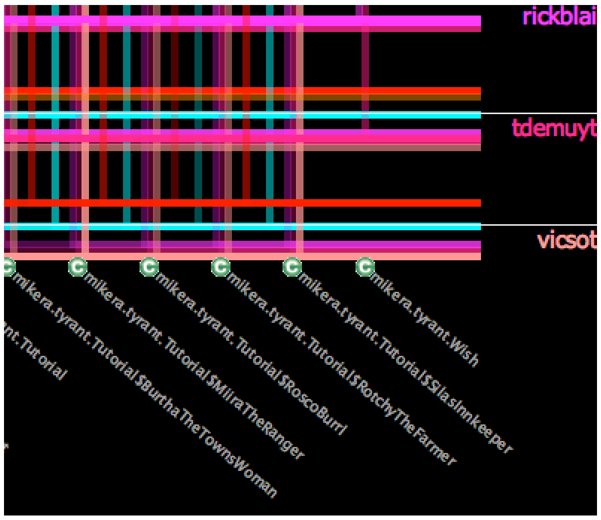
rickblai
tdemuyt
vicsot
mikera.tyrant.Tutorial
mikera.tyrant.Tutorial$BurthaTheTownsWoman
mikera.tyrant.Tutorial$MiraTheRanger
mikera.tyrant.Tutorial$RoscoBurl
mikera.tyrant.Tutorial$RotchyTheFarmer
mikera.tyrant.Tutorial$SilasInnkeeper
mikera.tyrant.Wish

Figure 1. Closeup of socio-technical dependencies in the "main" package of open-source Java project "Tyrant."

Ariadne draws connections from a dependent author to the code unit they are dependent upon and back to the author responsible for that code unit, and repeats for each set of socio-technical dependency information in the project. The color of each line (or dependency) denotes the directionality of the dependency and shares its color with the originating (dependent) author. An unfiltered overview (Figure 2) of the dependency information makes it possible to recognize patterns in the way developers call other developers' code, prominent code modules, and prominent authors even for a specific area of the code.

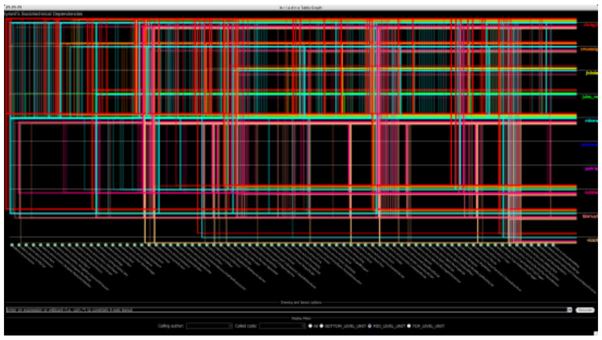

Figure 2. Developers' socio-technical relationships in the open-source Java project "Tyrant," revealing frequent use of modules by user "Chrisgri" (developer in red).

Filtering the overview by artifact reveals connections only from authors using that artifact. Managers and developers can focus on artifacts at different granularities that may be undergoing many changes in order to determine developers' progress, as indicated by our field studies [7]. Focusing on an artifact may allow managers and developers to locate other developers affecting or affected by changes to that artifact.

Using an additive approach, users can compare the calls on code units made by one author with those made by another author. The user can click on authors' names to reveal only their dependency information. Ariadne's visualization technique preserves the ease of identifying connections between authors found in simple social network graphs of developers. Looking at only the y-axis, users can readily determine the inbound and outbound connections between a project's developers. The presence of a color corresponding to an author's name indicates an outbound dependency, while the presence of other authors' colors indicates an inbound socio-technical dependency from those other authors. While Ariadne's visualization makes a significant departure from a more traditional graph-based approach, it does not eliminate the advantages of that method of data display.

3. Application of usability inspection methods

In order to assess the presentation, usability, and ease of learning of Ariadne's visualization, we evaluated it using Tufte's general principles [15, 16], the Heuristic Evaluation [11], the Cognitive Walkthrough [18], and the Cognitive Dimensions of

Notations [8]. We performed each inspection method with a team comprised of four colleagues. For the most part, they had no experience using the new visualization. This unfamiliarity helped us to identify problematic design assumptions about users' expectations and perceptions using the tool.

The combination of inspection methods allowed us to tease out the most important problems with the visualization. For example, the Cognitive Walkthrough, Tufte's principles, and the Cognitive Dimensions analyses pointed out problems with color choice. Possible solutions include using general color design guidelines [5] and selecting colors to support colorblind users [14]. The Heuristic Evaluation and Cognitive Dimensions revealed the potential need to allow users to undo certain filtering actions in order to trace back their steps, as well as the option to view different configurations of developers (into teams, for example) and system components. All three methods suggested the need to improve feedback, whether to indicate that specific dependencies have not been created, to display the calling code for a given dependency, or to show progress bars when the visualization undergoes a screen refresh.

Each usability inspection has its particular focus, so it is not surprising that the problems we found were problems the methods were intended to reveal. The Cognitive Walkthrough and Cognitive Dimensions focus on actions with the visualization that are mentally demanding. Accordingly, they revealed problems like keeping track of different colors and filters applied across use of the visualization. The Heuristic Evaluation, serving as a broad checklist of good usability principles, reinforced these findings and helped to identify improvements to be made in the future (e.g. help and documentation and correction of visual inconsistencies in filtered views).

4. Discussion and Related Work

The four analyses have allowed us to identify problems in the early stage of the development of our prototype of Ariadne before trials with human subjects. Eventually, we will run new trials with human subjects, though, generally speaking, human subject evaluations yield only performance data and not rationale that may affect design, especially in the early stages of design.

Some experimenters obtain rationale through Think Aloud methods. Nielsen and colleagues provide a recent, detailed discussion of applying this method and extensions to limit certain biases [12]. The rationale obtained in Think Aloud protocols is expensive in terms of obtaining subjects and performing the subsequent extensive analysis. The complexity and cost make it less appealing for early design.

New evaluation techniques for information visualization have recently emerged due to the limitations of current approaches to evaluation [1, 2, 14]. Some claim that evaluations targeted at visual interfaces test the wrong users [1, 14]. Unconventional interface components negatively impact user performance [1].

Testing the usability of visual tools with inspection methods is critical because assessment in real settings is a very rare possibility. At best, real employees can be brought in for laboratory experiments, and that is something we might do after further refinement of Ariadne. While we were not able to test the tools with real users, we were able to test it against tasks representative of real activities [7].

Ariadne's visual interface is not traditional. Typical user interface components like buttons and menus are not the primary focus. As such, in early design of the visualization, it is important to know whether users can overcome biases caused by the familiarity of traditional interface components [1]. The multiple inspection methods we applied here indicate that they can.

Visualization-specific heuristics can uncover issues that traditional usability heuristics may not [19]. The standard usability heuristics applied here were good enough to validate high-level perception-specific problems such as the use of color and detail shown in the interface. As we continue to refine Ariande with advanced information visualization-specific capabilities such as zooming and history views, we will look closer toward information visualization-specific heuristics such as those proposed by Ardito and colleagues [2]. Evaluation techniques that take into account the exploratory nature of users' tasks [14] will be useful in later stages of refinement of the tool.

5. Conclusions and future work

This paper described Ariadne, a visual software tool that translates technical dependencies in source-code to social dependencies between developers implementing that code. Ariadne has been motivated by our own empirical studies of software development projects and others'. The visualization is a revision of our original prototype [7].

We chose to evaluate the visual interface with usability inspection methods. To a degree, this approach is somewhat novel as these methods are normally applied to *user interface* components and not so often to workspace or *information interface* components. While every user interface might be called an information interface, our work provides a concrete example of amplifying the scope of usage of these methods to socio-technical information visualizations.

In conclusion, the inspection findings will lead us to improve the design of Ariadne before additional testing with human subjects. Moreover, the findings were sufficient to confirm the usefulness of these inspection methods in analyzing visual information interfaces and not just more traditional, menu-driven interfaces. Finally, inspection methods yield design explanations, answering questions about how and why an interface can be used to achieve its intended objectives.

6. Acknowledgments

This research is supported by the U.S. National Science Foundation under grants 0534775 and 0205724, and by an IBM Eclipse Technology Exchange grant. The authors gratefully acknowledge comments and encouragement from their colleague Steve Abrams and members of their department's "Continuous Coordination" group.

7. References

[1] K. Andrews, "Evaluating Information Visualisations", In *Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, ACM, New York, NY, 2006, pp.1-5.

[2] C. Ardito, P. Buono, M.F. Costabile, and R. Lanzilotti, "Systematic Inspection of Information Visualization Systems", In *Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, ACM, New York, NY, 2006, pp. 1-4.

[3] M. Cataldo, P.A. Wagstrom, J.D. Herbsleb, and K. Carley, "Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools", In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, ACM, New York, NY, 2006, pp. 353-362.

[4] W. Chisholm, et al., "W3C Web Content and Accessibility Guidelines 1.0", 1999. Available: http://www.w3.org/TR/WCAG10/

[5] M.E. Conway, "How Do Committees Invent?", *Datamation*, Thompson, 1968, 14 (4), pp. 28-31.

[6] B, Curtis, H. Krasner, and N. Iscoe, "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM*, ACM, New York, NY, 1988, 31 (11), pp. 1268-1287.

[7] C.R.B. de Souza, S. Quirk, E. Trainer, and D.F. Redmiles, "Supporting Collaborative Software Development through the Visualization of Socio-Technical Dependencies", In *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, ACM, New York, NY, 2007, pp. 147-156.

[8] T. Green, "Cognitive Dimensions of Notations", *People and Computers V Proceedings of HCI'89*, Cambridge University Press, New York, NY, 1990, pp. 443-460.

[9] J.D. Herbsleb and R.E. Grinter, "Architectures, Coordination, and Distance: Conway's Law and Beyond", *IEEE Software*, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 63-70.

[10] L. Jefferson and R. Harvey, "Accommodating Color Blind Computer Users", In *Proceedings of the 8th international ACM SIGACCESS Conference on Computers and Accessibility*, ACM, New York, NY, 2006, pp. 40-47.

[11] J.K. Nielsen, "Heuristic Evaluation", In *Usability Inspection Methods*, Wiley, New York, NY, 1994.

[12] J. Nielsen, T. Clemmensen, and C. Yssing, "Getting Access to What Goes on in People's Heads?: Reflections on the Think-Aloud Technique", In *Proceedings of the Second Nordic Conference on Human-Computer Interaction*, ACM, New York, NY, 2002, pp. 101-110.

[13] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", *Communications of the ACM*, ACM, New York, NY, 1972, 15 (12), pp. 1053-1058.

[14] C. Plaisant, "The Challenge of Information Visualization Evaluation", In *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM, New York, NY, 2004, pp. 109-116.]

[15] Tufte, E., *Beautiful Evidence*, Graphics Press, Cheshire, CT, 2006.

[16] Tufte, E., *Envisioning Information*, Graphics Press, Cheshire, CT, 1990.

[17] Wasserman, S. and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, Cambridge, UK, 1994.

[18] C.W. Wharton, J. Reiman, C. Lewis, and P. Polson, "The Cognitive Walkthrough Method: A Practitioner's Guide", In *Usability Inspection Methods*, Wiley, New York, NY, 1994.

[19] T. Zuk, L. Schlesier, P. Neumann, M.S. Hancock, and S. Carpendale, "Heuristics for Information Visualization Evaluation", In *Proceedings of the 2006 AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*, ACM, New York, NY, 2006, pp. 1-6.